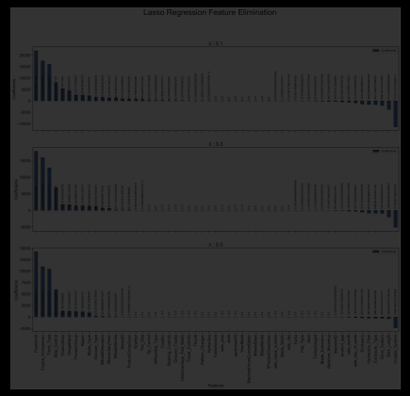
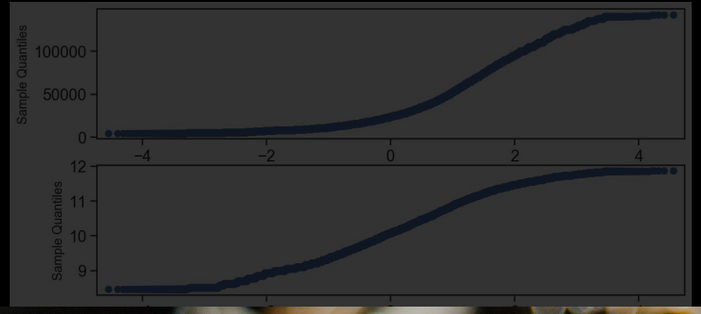
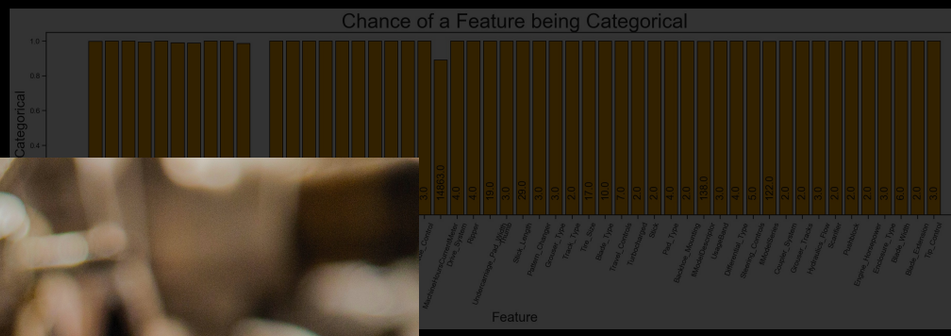
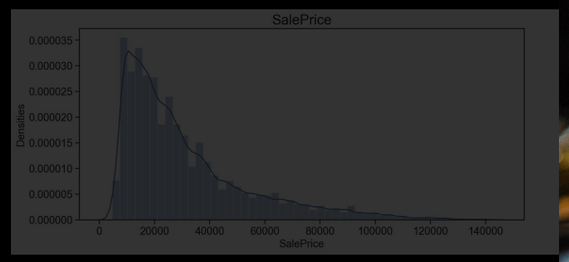
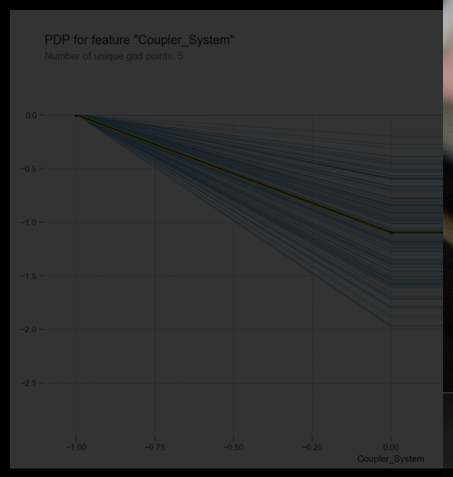








Weight	Feature
0.2010 ± 0.0019	Coupler_System
0.1681 ± 0.0068	product_age
0.1430 ± 0.0177	fiProductClassDesc
0.1364 ± 0.0104	ProductSize
0.0872 ± 0.0082	ModelID
0.0562 ± 0.0064	YearMade
0.0351 ± 0.0032	fiSecondaryDesc
0.0292 ± 0.0097	fiModelDesc
0.0187 ± 0.0010	sale_year
0.0182 ± 0.0095	fiBaseModel
0.0179 ± 0.0004	sale_week_number
0.0158 ± 0.0218	Enclosure
0.0138 ± 0.0004	sale_day
0.0131 ± 0.0004	state
0.0080 ± 0.0003	auctioneerID
0.0068 ± 0.0003	sale_day_of_week
0.0064 ± 0.0020	fiModelDescriptor
0.0063 ± 0.0011	fiModelSeries
0.0039 ± 0.0003	MachineHoursCurrentMeter
0.0032 ± 0.0016	Blade_Type
… 11 more …	



PREDICTING SALE PRICES

DATA PREP, FEATURE ENGINEERING, FEATURE SELECTION AND MODELLING PIPELINE

ACHINTYA GUPTA

SALE PREDICTION

Objective

- **Prediction :** To predict Sales Prices at an auction.

- **Modelling Pipeline :** To build a Modelling Pipeline which could,

    1. Ingest the data.
    2. Clean and transform the data.
    3. Run Base Modelling (evaluate using k-fold cross validation).
    4. Fine tune the models hyper parameters.
    5. Evaluate the selected model on test data.

- **Model explainability :** A simple walkthrough to the various libraries which could explain the "Black Box" models results and interpretations.

SALE PREDICTION

Process

Imports

Imports

Importing the necessary libraries that would be required in this entire process.
From data wrangling using *pandas* to data visualisations using *matplotlib*, *seaborn*.
Various libraries for Algorithms such as *sklearn* and Model Interpretability *ELI5*.

1 Imports

In [1]:

```
# General
import pandas as pd
import numpy as np
import os, ast
pd.set_option('display.max_colwidth', -1)
from tqdm import tqdm_notebook
import warnings
warnings.filterwarnings('ignore')
import pickle


# Visualisation
import seaborn as sns
from IPython.display import display_html
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot
from IPython.core.display import display, HTML


#Algorithms
import xgboost as xgb
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import Lasso
from sklearn.model_selection import RandomizedSearchCV, KFold, GridSearchCV
import pingouin as pg
from sklearn.metrics import r2_score

# Model Explainability
import shap
import eli5
import lime
from pdpbox import pdp, get_dataset, info_plots
from IPython.display import Image
from sklearn import tree
import pydotplus
```

executed in 3.60s, finished 23:00:07 2019-12-23

SALE PREDICTION

Process

Imports

Path / Variable Initialisation

[Config]

Path / Variable Initialisation

Initialising path,

- Where raw data sits
- Where prepared data will be stored
- Where Notebook Results will be saved

Initialising ***Target Variable***

Initialising ***Plotting Configurations***

2 Paths & Variable Initializations

```
1  ####### Path #######
2  root_path = '../'
3  data_path = root_path+'Data/'
4  raw_data_path = data_path+'Raw Data/'
5
6  complete_data_path = raw_data_path+'Data.csv'
7
8  prepared_data_path = data_path+'Prepared Data/'
9
10 notebook_results_path = root_path+'Notebook Results/'
11 os.makedirs(notebook_results_path, exist_ok=True)
12
13 ################VARIABLE INITIALISATIONS####################
14 TARGET_VARIABLE = 'SalePrice'
15
16 plt.rcParams['figure.dpi']     = 140
17 plt.rcParams['figure.figsize'] = (25,5)
```

SALE PREDICTION

Process

- Imports
- Path / Variable Initialisation [Config]
- Reading, Preparing & Splitting Data

Reading, Preparing and Splitting Data

Reading the raw data and analysing the individual columns, to gauge further into what sort of treatment would that column require. Once done with analysing columns individually some analysis of the NULL or Unique counts present in the dataset.

Preparing the data builds up on the preliminary analysis and the understanding of the data thus far, which could mean parsing few columns into date-time, dropping few irrelevant columns and converting categorical columns to their label encodings or one hot encoding.

SALE PREDICTION

Process

- Imports
- Path / Variable Initialisation [Config]
- Reading, Preparing & Splitting Data
- Feature Engineering

Feature Engineering

In Machine learning, Data prep constitutes around 60% of the efforts and about 20% of the efforts are utilised in thinking about the features that could be engineered from the data which is available to us. The remainder for Modelling!

Sometimes engineering new features which are not just explainable to the problem statement but are able to extract useful information can improve accuracy.

The categorical columns doesn't usually contain data in numerical form, the only form comprehensible to the machine, for that reason alone those categorical columns need to be converted into the either their label coding or One Hot Encodings.

5 Feature Engineering

In [ ]:
```
train_data_EDA = train_data.copy()
```
executed in 7ms, finished 16:58:54 2019-12-23

5.1 Adding Calendar Variables

In [ ]:
```
train_data_EDA['sale_year'] = train_data_EDA['saledate'].apply(lambda x :  x.year)
train_data_EDA['sale_month'] = train_data_EDA['saledate'].apply(lambda x :  x.month)
train_data_EDA['sale_day'] = train_data_EDA['saledate'].apply(lambda x :  x.day)
train_data_EDA['sale_day_of_week'] = train_data_EDA['saledate'].apply(lambda x :  x.dayofweek)
train_data_EDA['sale_week_number'] = train_data_EDA['saledate'].apply(lambda x :  x.week)
train_data_EDA['product_age'] = train_data_EDA.apply(lambda x: x.sale_year - x.YearMade if x.YearMade != -999 el

# Filtering out those DataPoints in which saledate was earlier than YearMade - Doesnt Make Sense, Right!
train_data_EDA = train_data_EDA[train_data_EDA['product_age'] >= 0]
```
executed in 21.8s, finished 16:59:17 2019-12-23

More Explanation on / from Slide 13

SALE PREDICTION

Process

Imports

Path / Variable Initialisation [Config]

Reading, Preparing & Splitting Data

Feature Engineering

Feature Selection

Feature Selection

From a plethora of features available to the modeller (i.e You), selecting which features to feed into the pipeline is a craftsmanship that only experience can teach, but few basic techniques are already there which could be leveraged for selecting features.

These technique can vary for the various types of the data, i.e to select a continuous feature w.r.t a continuous target variable correlation might be a good evaluator on the other hand Annova η^2 can be used to measure the association of a categorical column to that of an continuous target.

SALE PREDICTION

Process

- Imports
- Path / Variable Initialisation [Config]
- Reading, Preparing & Splitting Data
- Feature Engineering
- Feature Selection
- EDA

EDA

Exploratory Data Analysis, is one of those crucial steps where most of the intuition about modelling and new feature engineering / transformation required in the data available can be charted out.

There is no hard and fast rule which says that exploring and making intuitions should be done at this stage alone, most of the times I usually end up making my intuitions at the Data Prep & Feature Engineering Stage. Sometimes I have to back and forth between these steps.

6 EDA

In [ ]:
```
1  train_data_SF = train_data_EDA[selectedFeatures].copy()
```
executed in 20ms, finished 19:19:35 2019-12-17

6.1 Target Variable [...]

More Explanation on / from

Slide 19

SALE PREDICTION

Process

- Imports
- Path / Variable Initialisation [Config]
- Reading, Preparing & Splitting Data
- Feature Engineering
- Feature Selection
- EDA
- Modelling Pipeline

Modelling Pipeline

The concept of a Modelling Pipeline is as simple as, getting everything at one place in a structured fashion such that the pipeline functions can be reused to build models again and again when there is a

- Change in the Input (Data changes or selected features change)
- New model to be evaluated (Adding a new model into the pipeline or a new layer of modelling couch as Ensemble of Ensembles)

7 Modelling Pipeline

This class contains the entire pipeline for building base models:

Data Ingestion -> **Data Cleaning/Transformation** -> **Feature Engineering** -> **Modelling**

- **7.1 Read Data** [...]
- **7.2 Modelling Pipeline** [...]
- **7.3 Fine Tuning - HyperParameters** [...]
- **7.4 Predicting the Test Set** [...]

More Explanation on / from Slide 20

SALE PREDICTION

Process

Imports

Path / Variable Initialisation [Config]

Reading, Preparing & Splitting Data

Feature Engineering

Feature Selection

EDA

Modelling Pipeline

Model Interpretability

Model Interpretability

Until unless Simple Regression(Linear, Polynomial, GAM etc) models are used which can in their inherent nature are quite explainable, Explainability of a model becomes a huge blocker in convincing business to actually use the model in production.

Here I discuss few basic packages which can be used for unraveling few black box models such as tree based (Boosting Trees, XGBoost, CatBoost and even NN).

1) ELI5 ("Explain Like I'm 5” - The closest thing I could find)
2) SHAP (Shapely Additive Explanations)
3) LIME (Local Interpretable Model Agnostic Explanation)
4) PDP (Partial Dependence Plots)
5) ICE (Individual Conditional Explanations)
6) Tree Interpreter (If tree methods are being used)

More Explanation on / from Slide 24

SALE PREDICTION

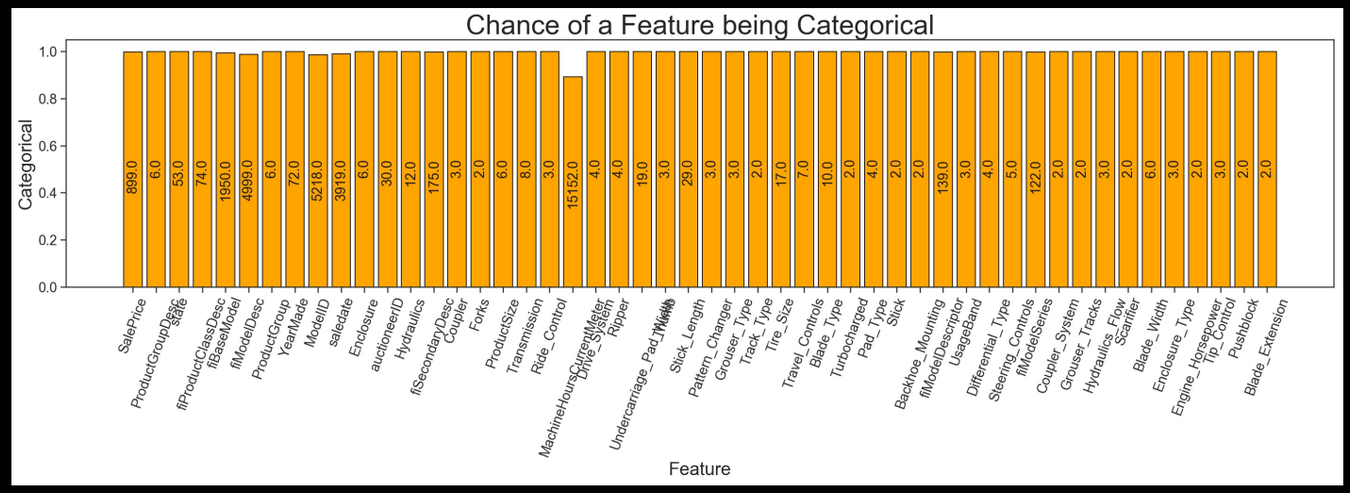
Understanding the data

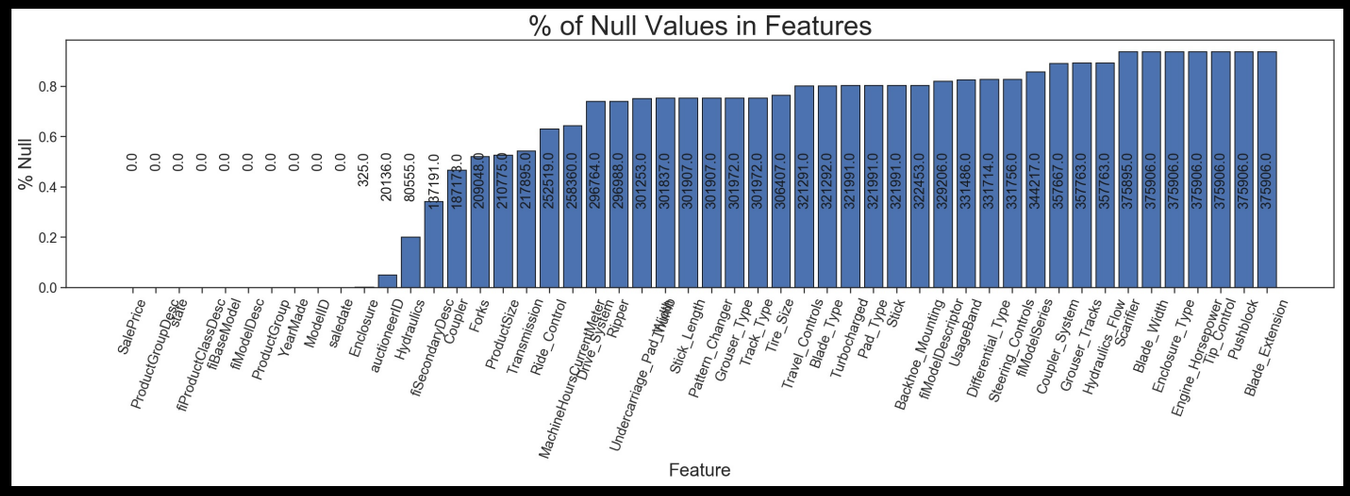
Variable	Description
SalesID	unique identifier of a particular sale of a machine at auction
MachineID	identifier for a particular machine; machines may have multiple sales
ModelID	identifier for a unique machine model (i.e. fiModelDesc)
datasource	source of the sale record; some sources are more diligent about reporting attributes of the machine than others.
auctioneerID	identifier of a particular auctioneer, i.e. company that sold the machine at auction. Not the same as datasource.
YearMade	year of manufacturer of the Machine
MachineHoursCurrentMeter	current usage of the machine in hours at time of sale (saledate); null or 0 means no hours have been reported for that sale
UsageBand	value (low, medium, high) calculated comparing this particular Machine-Sale hours to average usage for the fiBaseModel; e.g. 'Low' means this machine has less hours given it's lifespan relative to average of fiBaseModel.
Saledate	time of sale
Saleprice	cost of sale in USD
fiModelDesc	Description of a unique machine model (see ModelID); concatenation of fiBaseModel & fiSecondaryDesc & fiModelSeries & fiModelDescriptor
fiBaseModel	disaggregation of fiModelDesc
fiSecondaryDesc	disaggregation of fiModelDesc
fiModelSeries	disaggregation of fiModelDesc
fiModelDescriptor	disaggregation of fiModelDesc
ProductSize	Don't know what this is
ProductClassDesc	description of 2nd level hierarchical grouping (below ProductGroup) of fiModelDesc
State	US State in which sale occurred
ProductGroup	identifier for top-level hierarchical grouping of fiModelDesc
ProductGroupDesc	description of top-level hierarchical grouping of fiModelDesc
Drive_System	machine configuration; typcially describes whether 2 or 4 wheel drive
Enclosure	machine configuration - does machine have an enclosed cab or not
Forks	machine configuration - attachment used for lifting
Pad_Type	machine configuration - type of treads a crawler machine uses
Ride_Control	machine configuration - optional feature on loaders to make the ride smoother
Stick	machine configuration - type of control
Transmission	machine configuration - describes type of transmission; typically automatic or manual
Turbocharged	machine configuration - engine naturally aspirated or turbocharged
Blade_Extension	machine configuration - extension of standard blade
Blade_Width	machine configuration - width of blade
Enclosure_Type	machine configuration - does machine have an enclosed cab or not
Engine_Horsepower	machine configuration - engine horsepower rating
Hydraulics	machine configuration - type of hydraulics
Pushblock	machine configuration - option
Ripper	machine configuration - implement attached to machine to till soil
Scarifier	machine configuration - implement attached to machine to condition soil
Tip_control	machine configuration - type of blade control
Tire_Size	machine configuration - size of primary tires
Coupler	machine configuration - type of implement interface
Coupler_System	machine configuration - type of implement interface
Grouser_Tracks	machine configuration - describes ground contact interface
Hydraulics_Flow	machine configuration - normal or high flow hydraulic system
Track_Type	machine configuration - type of treads a crawler machine uses
Undercarriage_Pad_Width	machine configuration - width of crawler treads
Stick_Length	machine configuration - length of machine digging implement
Thumb	machine configuration - attachment used for grabbing
Pattern_Changer	machine configuration - can adjust the operator control configuration to suit the user
Grouser_Type	machine configuration - type of treads a crawler machine uses
Backhoe_Mounting	machine configuration - optional interface used to add a backhoe attachment
Blade_Type	machine configuration - describes type of blade
Travel_Controls	machine configuration - describes operator control configuration
Differential_Type	machine configuration - differential type, typically locking or standard
Steering_Controls	machine configuration - describes operator control configuration

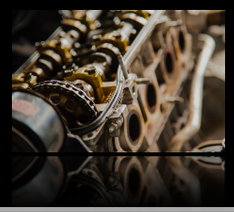
The data taken for this project is from Kaggle competition for *“Blue Book for Bulldozers”.*

The complete data that I considered for this problem statement was of : 401125 rows × 52 columns

← Target Column

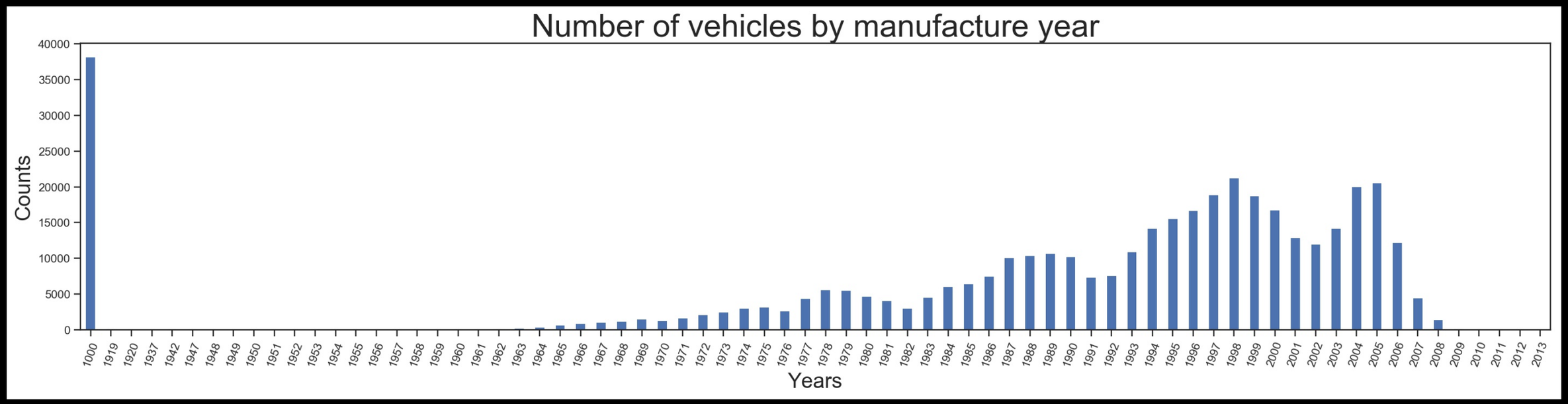






SALE PRICEDICTION

Understanding the data



In [176]:
```
## Analysing the data - You gotta go through each column one by one
idx = 0
```
executed in 3ms, finished 16:56:29 2019-12-23

In [177]:
```
colAnalysed = complete_data.columns[idx]
print('Column Being Analysed : ', colAnalysed)

helperFunctionsHandler.display_side_by_side([complete_data[[colAnalysed]].sample(7),
                                             complete_data[[colAnalysed]].describe()],
                                             captions = ['Raw Data', 'Descriptive Statistics'])
idx += 1

```
executed in 57ms, finished 16:56:29 2019-12-23

```
Column Being Analysed :  SalePrice
```

Raw Data

SalesID	SalePrice
1735746	14500
1796848	18000
2413133	46000
1640097	84000
1839870	38000
1589059	7850
1468915	57500

Descriptive Statistics

	SalePrice
count	401125
mean	31099.7
std	23036.9
min	4750
25%	14500
50%	24000
75%	40000
max	142000

- Target Column might be left skewed
- MachineID is not an unique identifier, as many vehicles were again auctioned
- ModelID is not an unique identifier, might be categorical
- datasource is an useless column, should be dropped
- auctioneerID is an categorical column
- There are 38185 data points in which YearMade is 1000 -- Junk Data
- Most of the data in 'MachineHoursCurrentMeter' is NaN -- replace it with -999
- UsageBand is categorical and should be ordered in order = ['High', 'Medium', 'Low]
- saledate is a date column. Parse it
- fiModelDesc, fiBaseModel, fiSecondaryDesc, fiModelSeries, fiModelDescriptor, fiProductClassDesc - All are categorical columns, and the missing values should be replaced wiith -999
- ProductSize is an categorical Size column
- state is an categorical column
- ProductGroup, ProductGroupDesc, Drive_System, Enclosure, Forks, Pad_Type, Ride_Control, Stick, Transmission, Turbocharged, Blade_Extension, Blade_Width, Enclosure_Type, Engine_Horsepower, Hydraulics, Pushblock, Ripper, Scarifier, Tip_Control, Tire_Size, Coupler, Coupler_System, Grouser_Tracks, Hydraulics_Flow, Track_Type, Undercarriage_Pad_Width, Stick_Length, Thumb, Pattern_Changer, Grouser_Type, Backhoe_Mounting, Blade_Type, Travel_Controls, Differential_Type, Steering_Controls, are categorical

Understanding the data is something which is very problem statement centric, it depends on the problem statement what exactly you want to analyse.

Analysing the Null and Unique values in a data frame can some time give good intuitions about the data and how the modelling might look like.

SALE PREDICTION

Feature Engineering

5 Feature Engineering

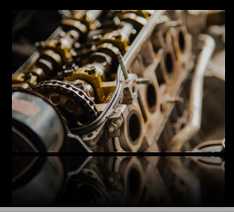
In [ ]:
```
1  train_data_EDA = train_data.copy()
```
executed in 7ms, finished 16:58:54 2019-12-23

5.1 Adding Calendar Variables

In [ ]:
```
1  train_data_EDA['sale_year'] = train_data_EDA['saledate'].apply(lambda x :  x.year)
2  train_data_EDA['sale_month'] = train_data_EDA['saledate'].apply(lambda x :  x.month)
3  train_data_EDA['sale_day'] = train_data_EDA['saledate'].apply(lambda x :  x.day)
4  train_data_EDA['sale_day_of_week'] = train_data_EDA['saledate'].apply(lambda x :  x.dayofweek)
5  train_data_EDA['sale_week_number'] = train_data_EDA['saledate'].apply(lambda x :  x.week)
6  train_data_EDA['product_age'] = train_data_EDA.apply(lambda x: x.sale_year - x.YearMade if x.YearMade != -999 el
7
8  # Filtering out those DataPoints in which saledate was earlier than YearMade - Doesnt Make Sense, Right!
9  train_data_EDA = train_data_EDA[train_data_EDA['product_age'] >= 0]
```
executed in 21.8s, finished 16:59:17 2019-12-23

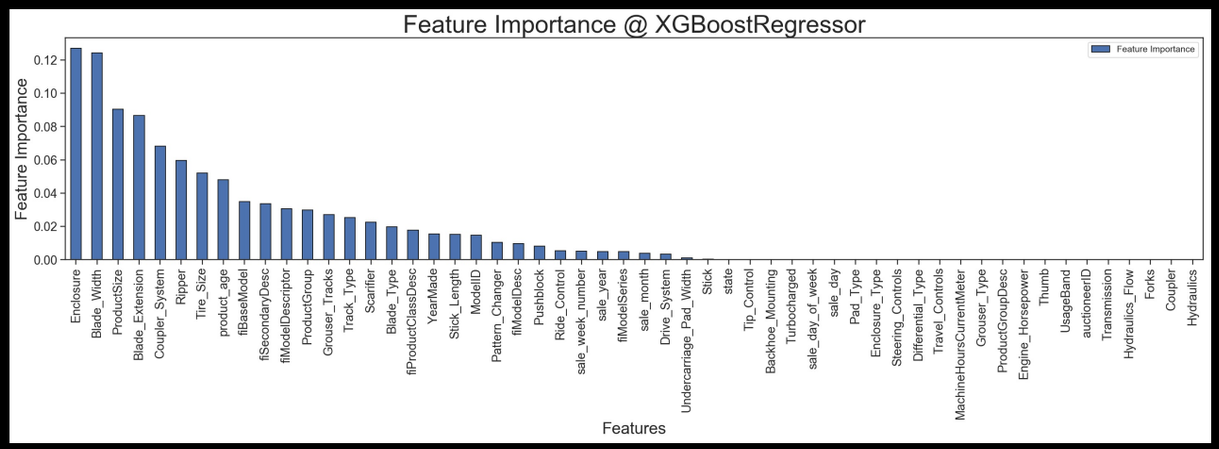
1. Preparing the data can involve removing those columns which have absolutely no relevance the problem at, for ex '*datasource*'.

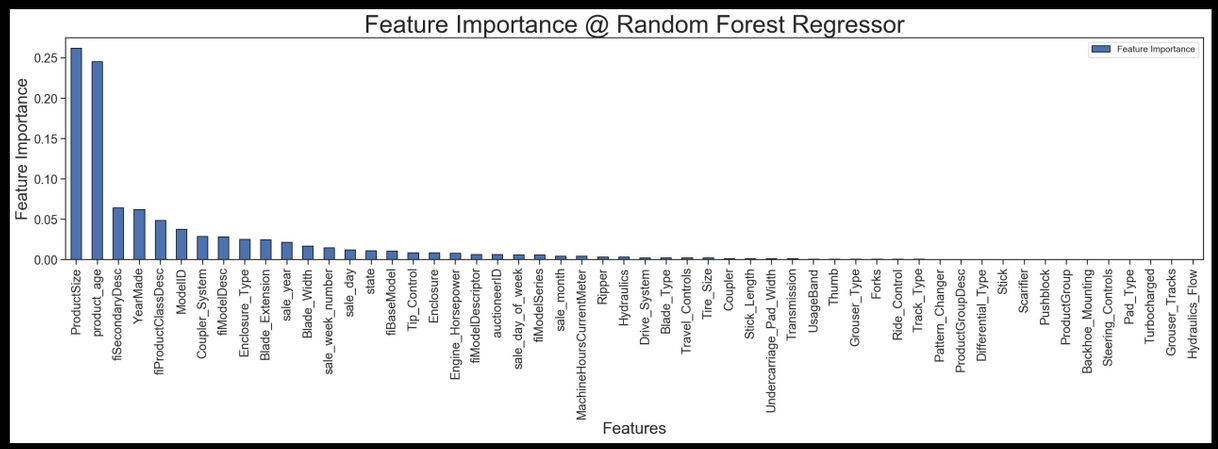
2. Now coming to feature engineering, I have dedicated a rather small chunk of my effort on the same, but quite honestly this is that aspect of Machine Learning which is probably where most of the creativity is involved and the *"Narnian Gateway"* for achieving higher score/accuracy for the model. Keeping that in mind I think this aspect deserves a separate post of itself. Although the features that I did engineer here were :
   - Calendar Variables
   - The age of the product - How old the product was at the time of auction from the year of manufacturing.



SALE PREDICTION

Feature Selection





Feature Importance

When there is a plethora of variables, it's usually a good call to remove few of those features which are not that relevant to the problem statement at hand. But how to figure those out?. In this project I have used following methodologies to select features, there are plenty more out there…

- Feature Importance - Just train a RandomForest or an XGBoost on the available data and see the feature importance of the variables

SALE PREDICTION

Feature Selection

	SalePrice
YearMade	0.220085
sale_year	0.039649
MachineHoursCurrentMeter	0.011550
sale_day	0.000552
ModelID	-0.034558
sale_week_number	-0.037413
sale_month	-0.037983
auctioneerID	-0.043810
sale_day_of_week	-0.052251
product_age	-0.250279

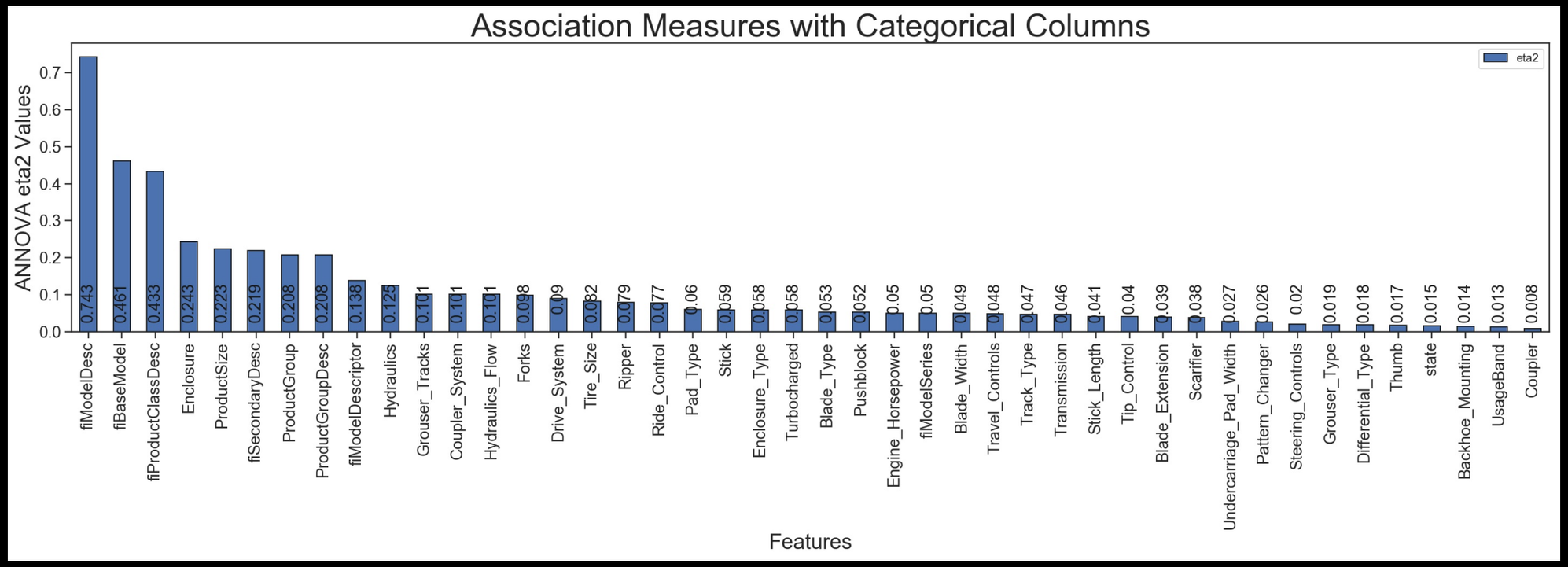
Correlation

When there is a plethora of variables, it's usually a good call to remove few of those features which are not that relevant to the problem statement at hand. But how to figure those out?. In this project I have used following methodologies to select features, there are plenty more out there…

- Correlation [continuous-continuous] - To calculate how much of the variance of a Target Variable is being explained by the feature, look at the Pearson Correlation of the same.

SALE PREDICTION

Feature Selection



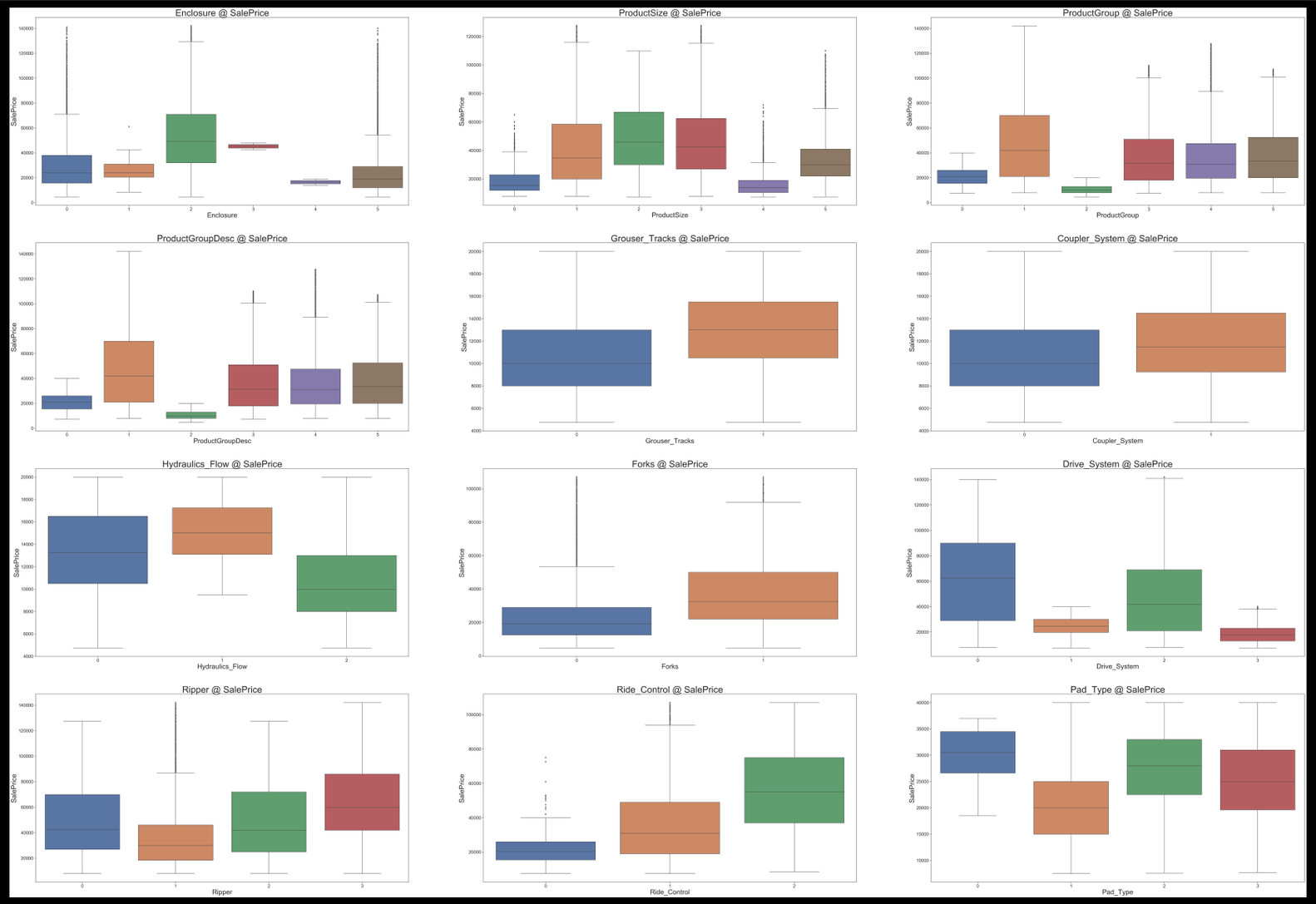
ANNOVA η^2 Association Value

When there is a plethora of variables, it's usually a good call to remove few of those features which are not that relevant to the problem statement at hand. But how to figure those out?. In this project I have used following methodologies to select features, there are plenty more out there…

- Annova **η^2** [continuous-discreet] - To gauge how associated a categorical feature is on Target Variable.

SALE PREDICTION

Feature Selection



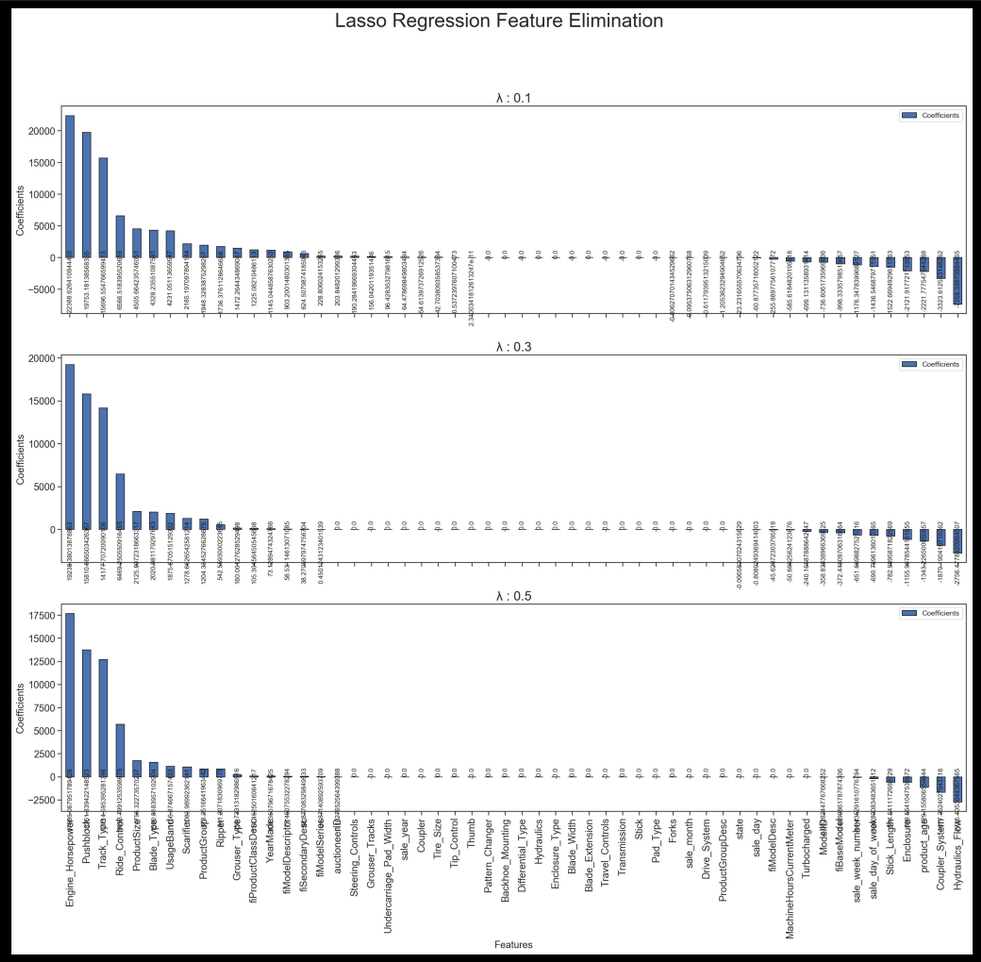
BoxPlots

When there is a plethora of variables, it's usually a good call to remove few of those features which are not that relevant to the problem statement at hand. But how to figure those out?. In this project I have used following methodologies to select features, there are plenty more out there…

- BoxPlots [continuous-discreet] - To filter for those features that have significant variation in their mean across classes wrt to Target variable.

SALE PREDICTION

Feature Selection



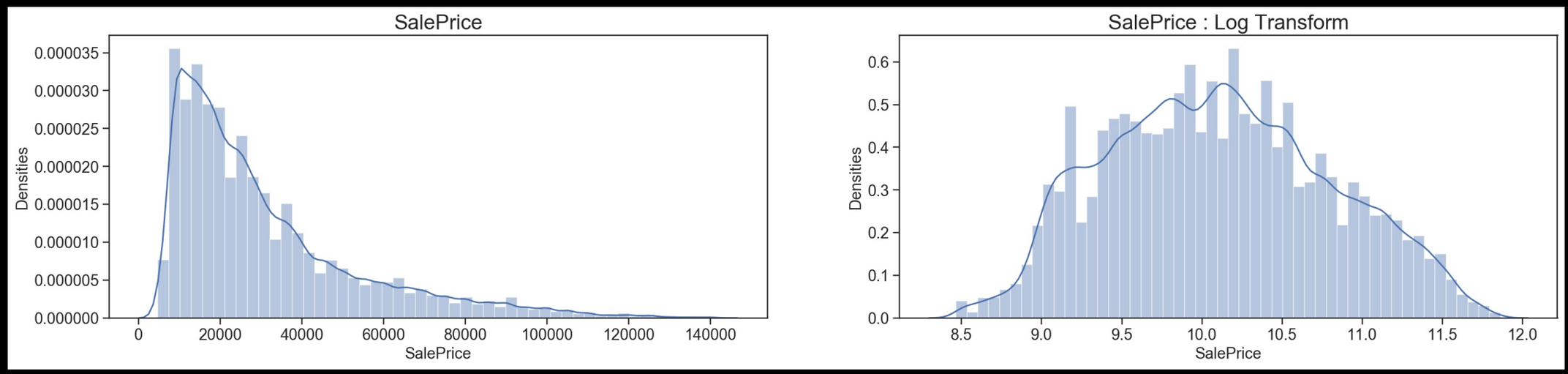
Lasso Regression Weights
varying Lambda

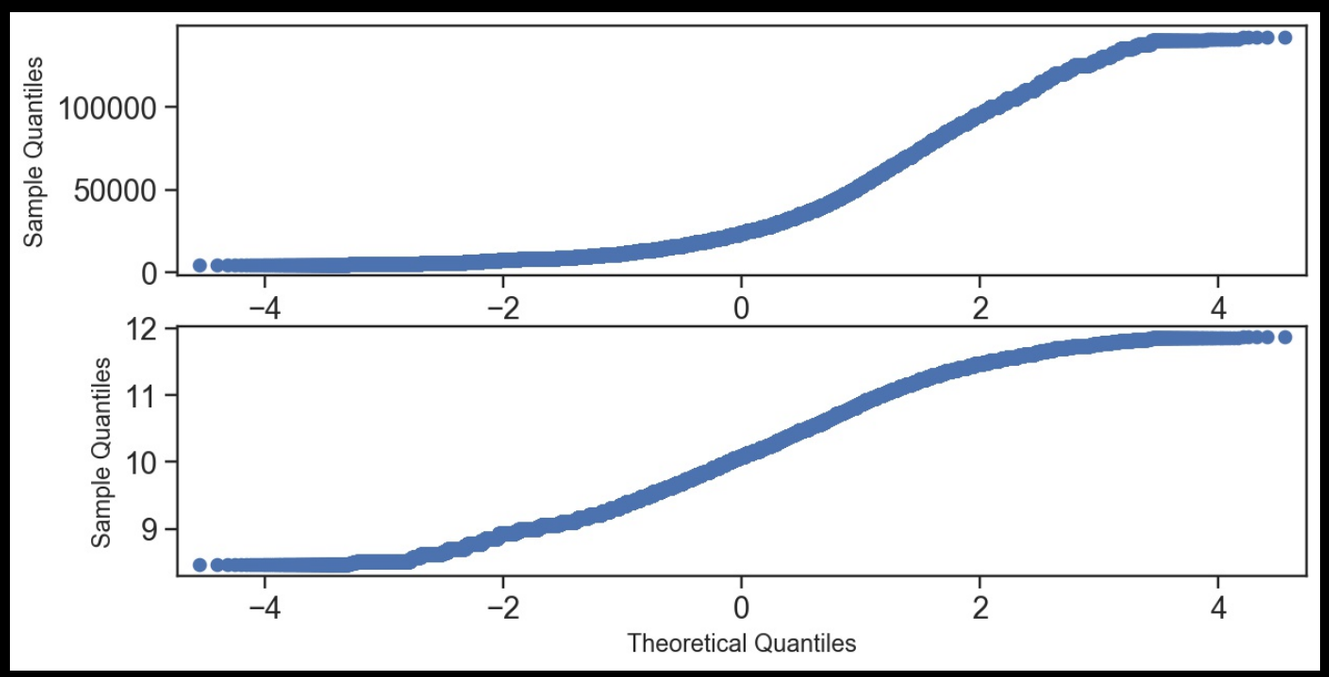
When there is a plethora of variables, it's usually a good call to remove few of those features which are not that relevant to the problem statement at hand. But how to figure those out?. In this project I have used following methodologies to select features, there are plenty more out there…

- Lasso Regression [Feature Elimination] - Eliminating features based on lasso regression.

SALE PREDICTION

EDA





Exploratory Data Analysis is usually takes up substantial effort in the Machine Learning exercise, although in this particular case, I restricted it to just analysing the Target Variable — 'SalePrice'.

Analysing the Target Variable in the raw form, it was clear that the data is skewed towards the left, applying a simple log transform on the same brought it closer to a Gaussian distribution plot and the Q-Q plot also seem to be *almost* linear to an ideal normal distribution

SALE PREDICTION

Modeling Pipeline

7 Modelling Pipeline

This class contains the entire pipeline for building base models:

Data Ingestion -> **Data Cleaning/Transformation** -> **Feature Engineering** -> **Modelling**

7.1 Read Data

In [215]:

```
complete_data = pd.read_csv(complete_data_path, index_col=0,low_memory=False)
split_ratio = 0.05
test_datapoints  = int(split_ratio * complete_data.shape[0])
TRAIN_DATA = complete_data[:-test_datapoints]
TEST_DATA = complete_data[-test_datapoints:]

selectedFeatures = pd.read_csv(notebook_results_path+'selectedFeatures.csv', index_col=0).values.ravel().tolist(
print('Train Data Shape : ', TRAIN_DATA.shape)
print('Test Data Shape : ', TEST_DATA.shape)
```

executed in 2.95s, finished 17:59:15 2019-12-23

```
Train Data Shape :  (381069, 52)
Test Data Shape :  (20056, 52)
```

7.2 Modelling Pipeline

In [210]:

```
class MLPipeline:

    def __init__(self, verbose = False):↔

    def ingestData(self, trainData, testData, targetVar):↔

    def cleanTransformData(self, selectedFeatures, fe_dependent_cols):↔

    def modelling(self, modelDictL1, metrics, splitter, backTransform = True, ↔

    def evaluateModel(self, metrics, savedModelPath = None):↔

    def calcMetrics(self, yTrue, yPred, _metrics):↔
```

executed in 78ms, finished 17:16:14 2019-12-23

7.3 Base Modelling [...]

7.4 Fine Tuning - HyperParameters [...]

7.5 Predicting the Test Set [...]

The idea behind **Modelling Pipeline** is that the we bring in all the requisite functions in one place so that ingesting, cleaning, transforming, base modelling on various models, hyper-parameter of those models, and once done selecting the model evaluate it on the test set.

SALE PREDICTION

Modeling Pipeline

7.3 Base Modelling

In [*]:

```
models_L1 = { 'XGBRegressor' : True,
              'CatBoostRegressor' : True,
              'LGBMRegressor' : True,
              'GradientBoostingRegressor' : True,
              'RandomForestRegressor' : True,
              'KNeighborsRegressor': True}

metrics = {'RMSLE':True,
           'MAPE':True,
           'MSE':True,
           'R2':True}

fe_dependent_cols = ['saledate', 'YearMade']

kfold_BM = KFold(n_splits=10, shuffle=True, random_state=1)

MLPipelineHandler = MLPipeline(verbose=True)
MLPipelineHandler.ingestData(TRAIN_DATA, TEST_DATA, TARGET_VARIABLE)
MLPipelineHandler.cleanTransformData(selectedFeatures, fe_dependent_cols)
SCORECARD = MLPipelineHandler.modelling(modelDictL1 = models_L1,
                                        metrics = metrics,
                                        splitter = kfold_BM)
```

execution queued 18:01:03 2019-12-23

```
Initialising Modelling Pipeline...

################### INGESTION ################
Ingesting Data into Pipeline...
##############################################


################### CLEANING #################
Cleaning Data...
Converting dtypes...
Filling missing values with -999...
Engineering Features...
Feature Transformation...
Converting dtypes...
Filling missing values with -999...
Engineering Features...
Feature Transformation...

Data Prepared!
##############################################


################## MODELLING #################


Training  XGBRegressor ...
        Evaluating for Fold 1
        Evaluating for Fold 2
        Evaluating for Fold 3
        Evaluating for Fold 4
        Evaluating for Fold 5
        Evaluating for Fold 6
        Evaluating for Fold 7
        Evaluating for Fold 8
        Evaluating for Fold 9
        Evaluating for Fold 10

RMSLE : 0.30357745580017526 | MAPE : 0.023257140852327437 | MSE : 0.09216010115005957 | R2 : 0.8070172236422802

BackTransformed-RMSLE : 11015.413827267217 | BackTransformed-MAPE : 664.3560401927249 | BackTransformed-MSE : 121344123.14474602 | BackTransformed-R2 : 0.7689280835225192


Training  CatBoostRegressor ...
        Evaluating for Fold 1
        Evaluating for Fold 2
        Evaluating for Fold 3
        Evaluating for Fold 4
        Evaluating for Fold 5
        Evaluating for Fold 6
        Evaluating for Fold 7
        Evaluating for Fold 8
        Evaluating for Fold 9
        Evaluating for Fold 10

RMSLE : 0.2456026660905764 | MAPE : 0.01826424298320438 | MSE : 0.060321000253312954 | R2 : 0.8736855146107303

BackTransformed-RMSLE : 8459.170941373633 | BackTransformed-MAPE : 508.6891612599351 | BackTransformed-MSE : 71560425.52436468 | BackTransformed-R2 : 0.8637277459661947
```

```
Training  LGBMRegressor ...
        Evaluating for Fold 1
        Evaluating for Fold 2
        Evaluating for Fold 3
        Evaluating for Fold 4
        Evaluating for Fold 5
        Evaluating for Fold 6
        Evaluating for Fold 7
        Evaluating for Fold 8
        Evaluating for Fold 9
        Evaluating for Fold 10

RMSLE : 0.2493709310316486 | MAPE : 0.01864028174545112 | MSE : 0.062186833279886854 | R2 : 0.8697761664043926

BackTransformed-RMSLE : 8698.39889612255 | BackTransformed-MAPE : 523.8171151220251 | BackTransformed-MSE : 75664809.93296461 | BackTransformed-R2 : 0.8559108112009722


Training  GradientBoostingRegressor ...
        Evaluating for Fold 1
        Evaluating for Fold 2
        Evaluating for Fold 3
        Evaluating for Fold 4
        Evaluating for Fold 5
        Evaluating for Fold 6
        Evaluating for Fold 7
        Evaluating for Fold 8
        Evaluating for Fold 9
        Evaluating for Fold 10

RMSLE : 0.3031737122688127 | MAPE : 0.023211846582722857 | MSE : 0.09191556470089533 | R2 : 0.807530840804629

BackTransformed-RMSLE : 11003.52580259459 | BackTransformed-MAPE : 663.1908871379671 | BackTransformed-MSE : 121082357.69975547 | BackTransformed-R2 : 0.7694281226422579


Training  RandomForestRegressor ...
        Evaluating for Fold 1
        Evaluating for Fold 2
        Evaluating for Fold 3
        Evaluating for Fold 4
        Evaluating for Fold 5
        Evaluating for Fold 6
        Evaluating for Fold 7
        Evaluating for Fold 8
        Evaluating for Fold 9
        Evaluating for Fold 10

RMSLE : 0.21703780217080176 | MAPE : 0.015367777080302281 | MSE : 0.04710675376829128 | R2 : 0.9013559287445311

BackTransformed-RMSLE : 7285.101091028926 | BackTransformed-MAPE : 427.0531064475372 | BackTransformed-MSE : 53076618.10543309 | BackTransformed-R2 : 0.8989316934518614

##############################################
```

SALE PREDICTION

Modeling Pipeline

8.4 Fine Tuning - HyperParameters

In [7]:

```
models_L1 = {'RandomForestRegressor' : True}
finetune_Params = {'RandomForestRegressor' : {'n_estimators' : 100, # Number of estimators
                                              'criterion' : 'mse',
                                              'max_depth' : 5, # Maximum Depth of the trees to build
                                              'max_features' : 'auto' # Can take 'auto', 'sqrt', 'log2'
                                              }
                   }

metrics = {'RMSLE':True,
           'MAPE' :True,
           'MSE'  :True,
           'R2'   :True}

fe_dependent_cols = ['saledate', 'YearMade']

kfold_BM = KFold(n_splits=2, shuffle=True, random_state=1)

MLPipelineHandler = MLPipeline(verbose=True)
MLPipelineHandler.ingestData(TRAIN_DATA,
                             TEST_DATA,
                             TARGET_VARIABLE)

MLPipelineHandler.cleanTransformData(selectedFeatures,
                                     fe_dependent_cols)

SCORECARD = MLPipelineHandler.modelling(modelDictL1 = models_L1,
                                        metrics = metrics,
                                        splitter = kfold_BM,
                                        finetune_mode = True,
                                        finetune_paramdict = finetune_Params,
                                        finetune_savemodel=True,
                                        finetunemodel_savepath=notebook_results_path)

```

executed in 1m 17.7s, finished 19:34:49 2019-12-23

```
Initialising Modelling Pipeline...

#################### INGESTION ################
Ingesting Data into Pipeline...
###############################################



#################### CLEANING #################
Cleaning Data...
Converting dtypes...
Filling missing values with -999...
Engineering Features...
Feature Transformation...

Data Prepared!
###############################################



################## MODELLING ################


Training  RandomForestRegressor ...
############## BUILDING FINAL MODEL ############
Loading Model..
Model saved at : ../Notebook Results/RandomForestRegressor-finetunedModel.pkl
###############################################
```

SALE PREDICTION

Modeling Pipeline

8.5 Predicting the Test Set

In [*]:

```
models_L1 = {'RandomForestRegressor' : True}

metrics = {'RMSLE': True,
           'MAPE' : True,
           'MSE'  : True,
           'R2'   : True}

fe_dependent_cols = ['saledate', 'YearMade']

MLPipelineHandler = MLPipeline(verbose=True)
MLPipelineHandler.ingestData(TRAIN_DATA,
                             TEST_DATA,
                             TARGET_VARIABLE)

MLPipelineHandler.cleanTransformData(selectedFeatures,
                                     fe_dependent_cols)

model = MLPipelineHandler.evaluateModel(metrics = metrics,
                                        savedModelPath = '../Notebook Results/RandomForestRegressor-finetunedMod
```

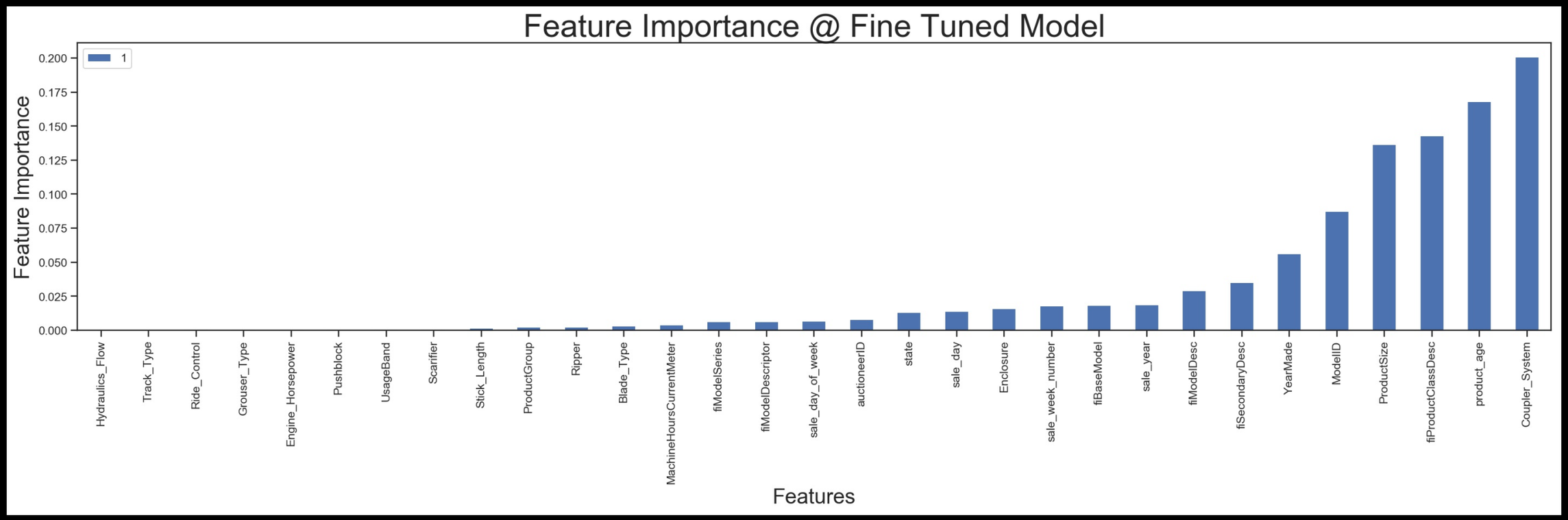
execution queued 19:35:50 2019-12-23

```
Initialising Modelling Pipeline...

#################### INGESTION ################
Ingesting Data into Pipeline...
###############################################



#################### CLEANING #################
Cleaning Data...
Converting dtypes...
Filling missing values with -999...
Engineering Features...
Feature Transformation...
Converting dtypes...
Filling missing values with -999...
Engineering Features...
Feature Transformation...

Data Prepared!
###############################################


Test Results :
```

RMSLE : 0.4696427915839575 | MAPE : 0.03623360310026743 | MSE : 0.22056435168677252 | R2 : 0.5969918478672901

SALE PREDICTION

Model Interpretability



Feature Importance

Feature importance for a model

SALE PREDICTION

Model Interpretability

Weight	Feature
0.2010 ± 0.0019	Coupler_System
0.1681 ± 0.0068	product_age
0.1430 ± 0.0177	fiProductClassDesc
0.1364 ± 0.0104	ProductSize
0.0872 ± 0.0082	ModelID
0.0562 ± 0.0064	YearMade
0.0351 ± 0.0032	fiSecondaryDesc
0.0292 ± 0.0097	fiModelDesc
0.0187 ± 0.0010	sale_year
0.0182 ± 0.0095	fiBaseModel
0.0179 ± 0.0004	sale_week_number
0.0158 ± 0.0218	Enclosure
0.0138 ± 0.0004	sale_day
0.0131 ± 0.0004	state
0.0080 ± 0.0003	auctioneerID
0.0068 ± 0.0003	sale_day_of_week
0.0064 ± 0.0020	fiModelDescriptor
0.0063 ± 0.0011	fiModelSeries
0.0039 ± 0.0003	MachineHoursCurrentMeter
0.0032 ± 0.0016	Blade_Type
… 11 more …	

ELI5 - Global Interpretation

Interpreting the model globally on how important a particular feature is for the model with respective confidence interval.

y (score **11.175**) top features

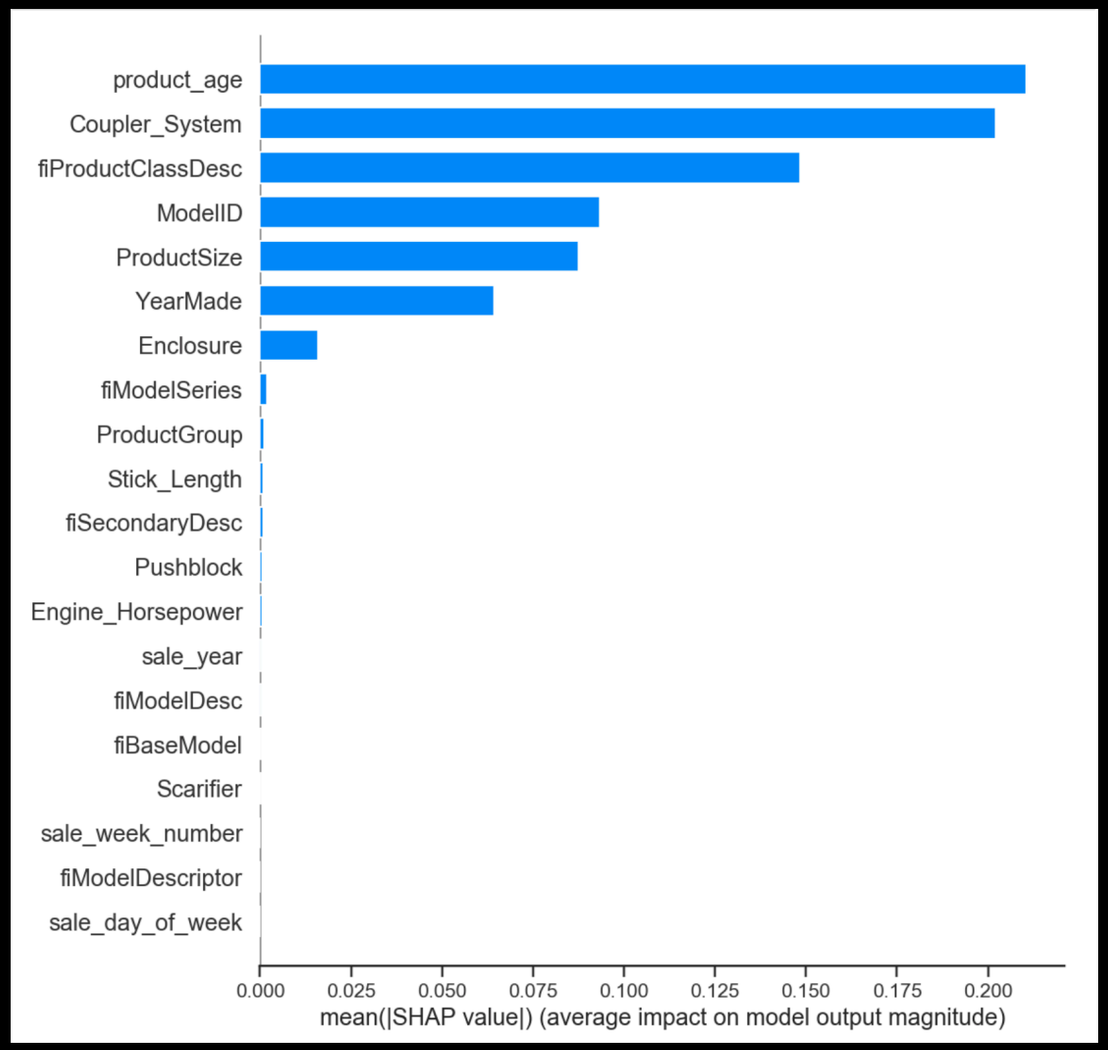
Contribution[?]	Feature
+10.102	<BIAS>
+0.316	ProductSize
+0.310	product_age
+0.203	fiProductClassDesc
+0.107	Coupler_System
+0.096	sale_week_number
+0.091	fiModelSeries
+0.065	Enclosure
+0.018	fiBaseModel
+0.016	fiSecondaryDesc
+0.015	fiModelDesc
+0.010	sale_day
+0.009	Stick_Length
+0.007	state
+0.006	sale_day_of_week
+0.004	Ripper
+0.003	auctioneerID
+0.002	Grouser_Type
+0.002	fiModelDescriptor
+0.001	ProductGroup
+0.001	Track_Type
+0.000	Blade_Type
+0.000	Ride_Control
-0.001	Scarifier
-0.002	UsageBand
-0.003	Pushblock
-0.003	Engine_Horsepower
-0.008	MachineHoursCurrentMeter
-0.017	YearMade
-0.041	sale_year
-0.133	ModelID

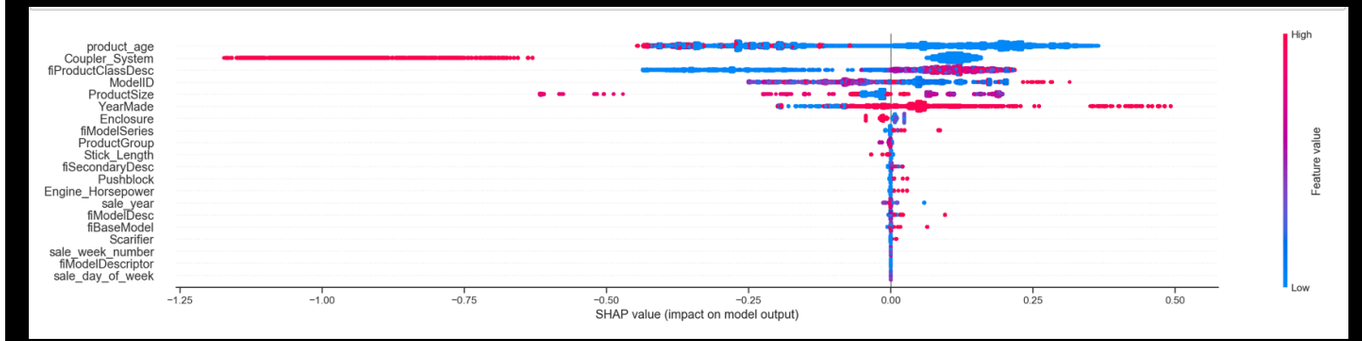
ELI5 - Local Interpretation

Interpreting the model for individual datapoint a particular feature is for the model.

SALE PREDICTION

Model Interpretability





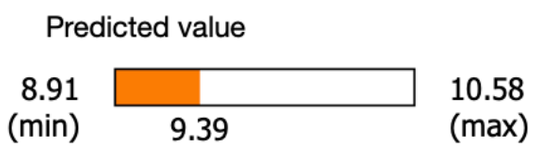
SHAP - Global

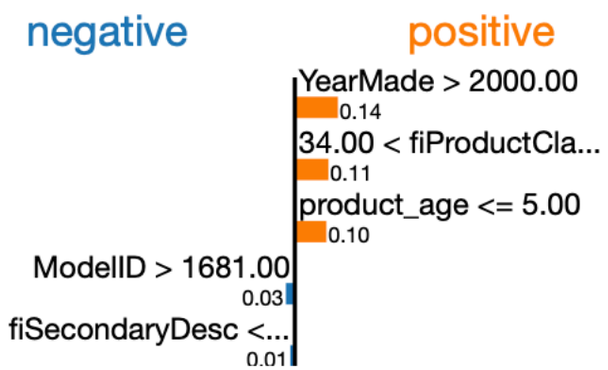
SHAP - Local

SALE PREDICTION

Model Interpretability

```
Intercept 9.045397386616157
Prediction_local [9.35118702]
Right: 9.385470803567575
```



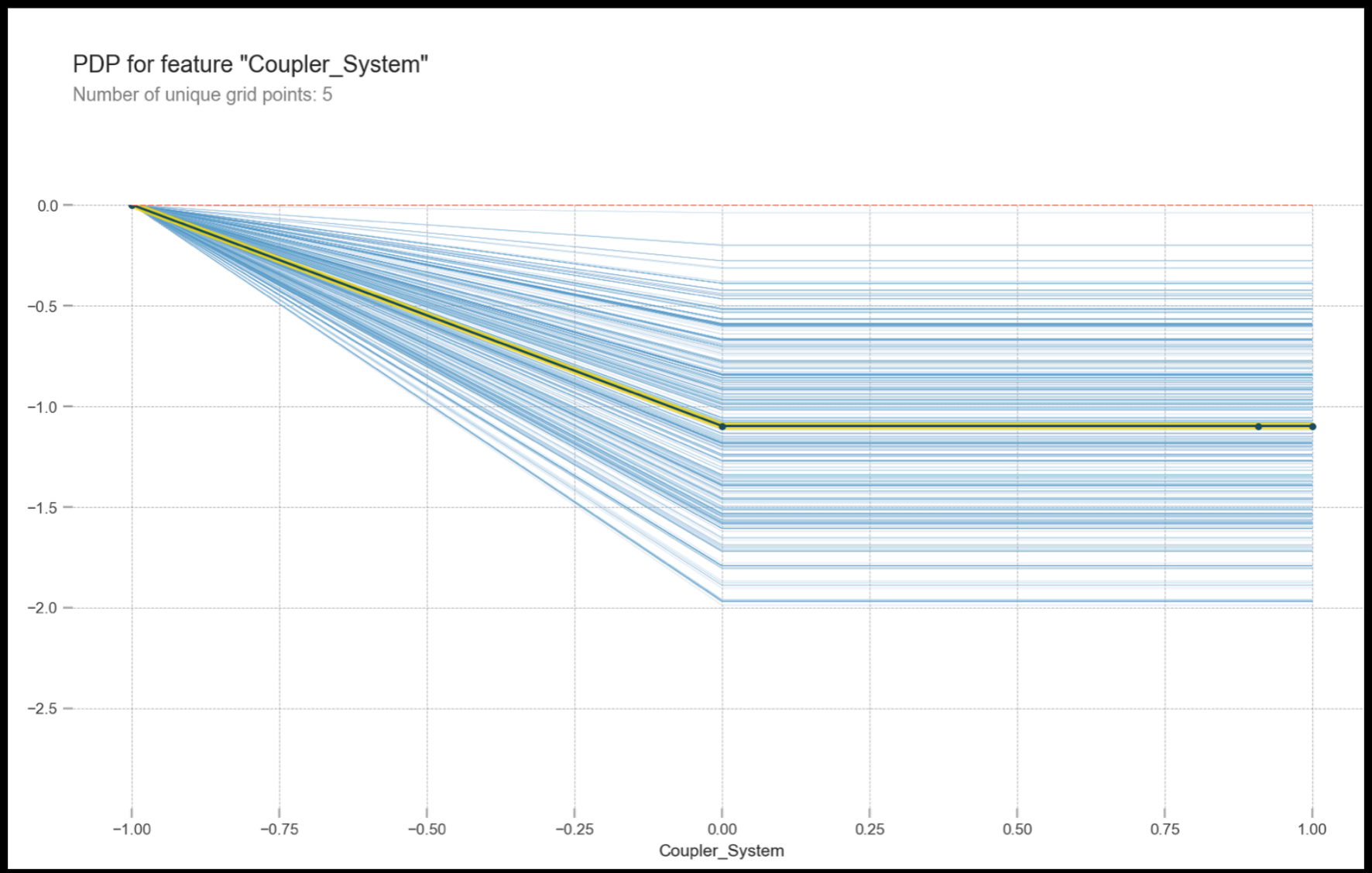


Feature	Value
YearMade	2006.00
fiProductClassDesc	45.00
product_age	5.00
ModelID	2036.00
fiSecondaryDesc	-1.00

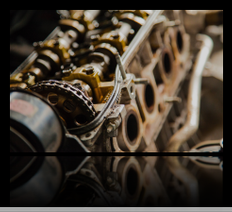
LIME

SALE PREDICTION

Model Interpretability

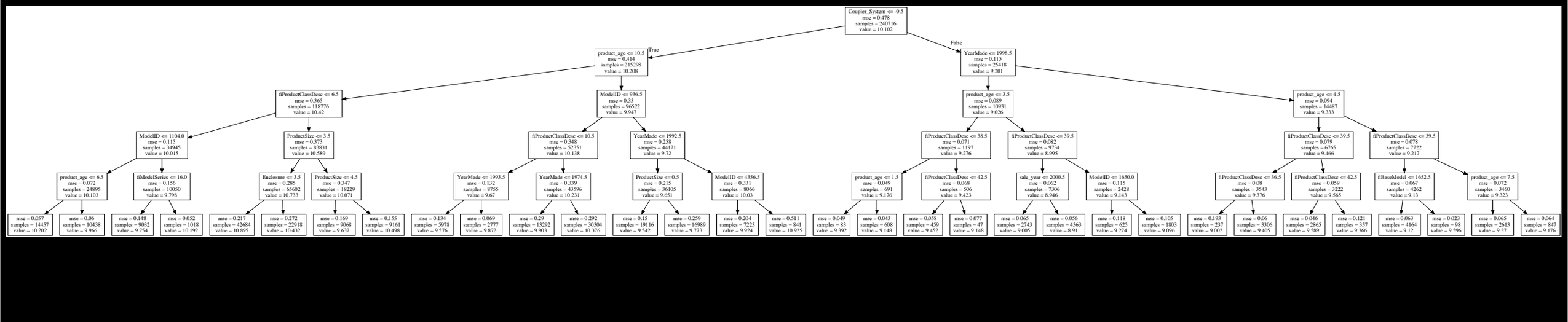


PDP & ICE



SALE PRICE PREDICTION

Model Interpretability



Decision Tree Visualisation